WWW.SCIAM-MAGAZINE.NAROD.RU

Гольф для любителей программирования

А.К. ДЬЮДНИ

В ОБЫЧНОМ гольфе игрок, используя различные биты, пытается загнать мяч в лунку, скрытую в траве на газоне в несколько акров с прудами, деревьями и ямками. В минигольф играют одной битой на гораздо меньшем игровом поле с небольшими препятствиями в виде кочек, тоннелей, вращающихся лопастей и качающихся маятников. Тому же, кто играет в микрогольф, не нужна никакая бита, а игровым полем им служит экран микрокомпьютера. В этом случае попасть в лунку с первого «удара» тоже нелегко; здесь имеются свои препятствия, в том числе и такие, которые противоречат законам физики.

Хотя в данном случае играть можно, сидя в кресле, игрок все же может ударять по мячу электронной битой и даже получать удовольствие от такого элемента в игре, которого нет в гольфе на естественных площадках, а именно программирования. Многие ли из нас способны разметить игровое поле, подготовить его к игре и сыграть матч — и все это в один день? Это не по плечу даже Джеку Никлаусу (чемпиону гольфа — *Ред.*) — если, конечно, он не умеет программировать. На самом деле, для того чтобы построить площадку для микрогольфа, ему совсем не обязательно быть профессиональным программистом.

Я укажу три варианта размещения трех лунок: один для начинающих игроков и два для более опытных. К концу дня каждый, кто играет по установленным правилам, овладеет игрой в микрогольф в той степени, в какой ему позволят собственные способности.

Простейший вариант игры представлен на рисунке внизу. Здесь программа, которую я назвал HOLE IN ONE, изображает всего одну лунку на прямоугольном зеленом поле. Бита выглядит как короткий отрезок прямой линии; она занесена над мячом, расположенным на краю поля. Игрок устанавливает биту в определенное положение и выбирает угол ее наклона так, чтобы послать мяч к цели — лунке, находящейся на другом краю игрового поля.

Точный удар на игровом поле для гольфа, построенном программой HOLE IN ONE

В простейшей версии игры, HOLE IN ONE, игрок наносит только один удар и либо попадает, либо промахивается. Если мяч не попадает в лунку, он уходит за край поля и исчезает с экрана, будто его и не было, он навсегда теряется в памяти компьютера. Но и в этой упрощенной версии игра довольно увлекательна и вызывает некоторый азарт.

Написать программу HOLE IN ONE и воспользоваться ею может любой рядовой программист-любитель; для него это будет своеобразным трамплином для перехода к более сложным версиям игры. Некоторые фрагменты из HOLE IN ONE можно вставить в более сложные программы, которые я назвал BIRDIE и EAGLE; они приведены на с. 87 и 88.

В программе BIRDIE предусмотрено препятствие, расположенное вблизи лунки; это круговая зона, называемая ловушкой. Попав в нее, мяч изменяет направление и скорость движения совершенно непредсказуемым образом. Еще более коварные препятствия имеет программа EAGLE, но о них поговорим позже.

Примерно раз в год я обычно пытался давать в одной из статей более подробное описание программ для тех неуверенных в себе или начинающих программистов, которым требуется немного дополнительной информации, чтобы начать работать самостоятельно. В приведенном ниже описании программы HOLE IN ONE я воспользовался старым проверенным методом и начал с подробного описания алгоритма. Такой способ позволяет быстрее перейти к фрагментам настоящей программы. Читателям, которые сумеют во всем этом разобраться, останется лишь нажимать на клавиши, чтобы принять участие в открытом чемпионате на приз нашего журнала.

Программа HOLE IN ONE сначала показывает игровое поле и лунку и затем предлагает игроку прицелиться и нанести удар битой. Затем программа многократно «перерисовывает» мяч, катящийся по электронному газону, возможно, в лунку.

Руководствуясь этими правилами, можно уже в общих чертах составить алгоритм программы HOLE IN ONE: нарисовать площадку, предложить играющему нанести удар, многократно продвинуть мяч (создать своеобразную мультипликацию по направлению и скорости биты) и определить, попадет ли мяч в лунку. Потом алгоритм можно уточнить, чтобы перейти к написанию программы. На-

WWW.SCIAM-MAGAZINE.NARODU.RU

чнем с того, что исходное положение игры можно, например, представить последовательным изображением сначала прямоугольной площадки, затем мяча, лунки и биты. Эти шаги можно легко запрограммировать.

Для определенности будем предполагать, что программа пишется на языке Бейсик для персонального компьютера фирмы IBM. (Не огорчайтесь, если у вас другой компьютер. Программу легко приспособить к любой другой системе.) Для еще большей определенности будем предполагать, что экраны всех компьютеров имеют в ширину по крайней мере 300 пикселов (точечных элементов изображения) и 200 пикселов в высоту. Все расстояния на экране должны измеряться от начала координат, которое находится в левом верхнем углу экрана.

Для экрана размером 300 × 200 пикселов вполне подойдет поле в виде прямоугольника 240 пикселов в длину и 160 пикселов в высоту. Чтобы расположить игровое поле примерно в центре экрана, программа HOLE IN ONE смещает его на 30 пикселов от левого края и на 20 пикселов от верхнего края экрана. Короче говоря, горизонтальные координаты поля меняются в диапазоне от 30 до 270, а вертикальные — от 20 до 180. Зеленое поле принимает свои очертания на экране по ходу выполнения следующих инструкций:

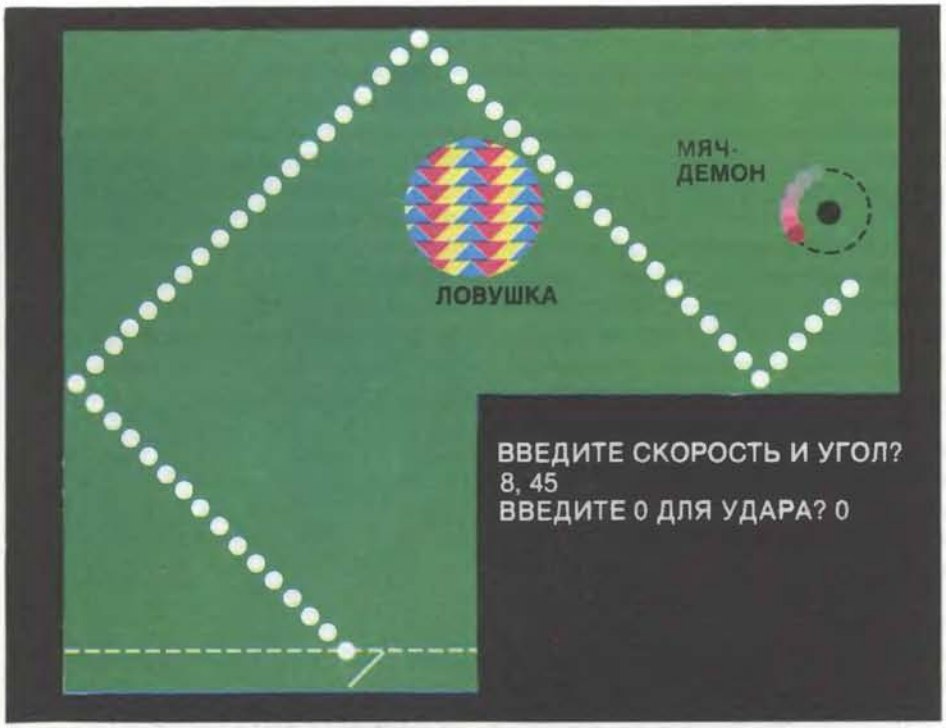
```
10 CLS
20 LINE (30, 20) – (270, 20)
30 LINE (30, 180) – (270, 180)
40 LINE (30, 20) – (30, 180)
50 LINE (270, 20) – (270, 180)
```

Заметим, что, нумеруя инструкции Бейсика в возрастающем (но не обязательно равномерно) порядке, программист задает точную последовательность, в которой компьютер должен выполнять эти инструкции. Первая команда CLS очищает экран от любых изображений, оставшихся на нем от выполнения предыдущих действий. Этой командой обычно пользуются перед тем, как запустить программу.

Программа HOLE IN ONE представляет лунку в виде кружочка у дальнего конца прямоугольника. Центр его находится в 240 пикселах от левого края и в 100 пикселов от верхнего края. Следовательно, инструкция Бейсика, которая должна нарисовать лунку, выглядит так:

```
60 CIRCLE (240, 100), 5, 1
```

Центр кружочка располагается в точке с координатами (240, 100), а его радиус равен 5 пикселам. Единица в конце строчки указывает цвет — белый.



Ловушка и мяч-демон значительно затрудняют игру

Чтобы игра не казалась слишком однообразной, программа помещает мячик каждый раз в случайно выбранной точке на исходной линии, которая удалена на 30 пикселов от левого края игровой площадки. Поскольку по ходу выполнения программы положение мяча изменяется, она следит за его координатами с помощью специально созданных для этого переменных *XBALL* и *YBALL*.

```
70 XBALL = 60
80 YBALL = 160 * RND + 20
90 CIRCLE (XBALL, YBALL), 4, 1
```

Строка 70 ограничивает координаты мяча исходной прямой линией. В строке 80 выбирается случайное число RND в диапазоне между 0 и 1, приводит его к масштабу от 0 до 160 и затем прибавляет к результату 20. Это означает, что значение *YBALL* попадет в какую-то точку вдоль исходной линии. В строке 90 находится команда, рисующая мяч с центром в точке (*XBALL, YBALL*). Радиус мяча равен 4, так чтобы он умещался в лунке с радиусом 5.

Отрезок прямой линии позади мяча представляет биту, или точнее, ее ударную часть. Координаты ее неподвижного положения задаются следующим образом:

```
100 LINE (54, YBALL + 8) –
    – (54, YBALL – 8)
```

Эта вертикальная линия касательна к левому краю мяча, а ее длина составляет 16 пикселов.

Следующий шаг алгоритма программы — «предложить играющему сделать удар». Прежде чем продолжить составление программы, уточним, что «предложить» — означает не только вывести соответствующее сообщение на экран, но также получить информацию об ориентации биты и скорости замаха. Нужно также перерисовать биту и изобразить ее в новом положении, так чтобы играющий смог оценить удар визуально.

Дополним поэтому программу HOLE IN ONE двумя строками, с помощью которых и выводится сообщение и принимаются необходимые входные данные:

```
110 PRINT «ВВЕДИТЕ СКОРОСТЬ И УГОЛ»
120 INPUT SPEED, AIM
```

В переменной *SPEED* (скорость) хранится расстояние в пикселах, которое проходит мяч всякий раз, когда программа заново вычисляет его координаты в процессе полета.

Переменная *AIM* (цель) хранит направление движения мяча. Многие, видимо, согласятся, что направление лучше всего выражать в градусах, которые и записываются в *AIM*. Поэтому программа HOLE IN ONE принимает значения угла в диапазоне от −90° до +90°, отсчитываемого от горизонтальной линии. Максимальное и минимальное значения угла соответствуют ударам, посылающим мяч прямо вверх или вниз.

К сожалению, в большинстве распространенных версий языка Бейсик

WWW.SCIAM-MAGAZINE.NARODRU

углы измеряются не в градусах, а в единицах, называемых радианами. Поэтому в программу необходимо вставить небольшое вычисление, с помощью которого градусы переводятся в радианы.

```
130 RADAIM = (AIM * 3.1416)/180
```

В основе этого преобразования лежит условие: 180° соответствуют π радиан.

Значения переменных *RADAIM* и *SPEED* можно подставить в формулу, с помощью которой определяются положение и ориентация биты. Программа HOLE ON ONE требует, чтобы в соответствии с формулой на экране изображалась бита, занесенная для удара, после которого мяч либо попадет в лунку, либо улетит за пределы экрана.

Формула помещает центр биты за мячом по линии удара и на расстоянии, пропорциональном значению переменной *SPEED*. В случае когда значение *SPEED* равно нулю, бита должна касаться мяча, т. е. она должна находиться по крайней мере на расстоянии трех пикселов слева от центра мяча. Поэтому координата *x* центра биты определяется вычитанием величины (3 + *SPEED*), умноженной на косинус *RADAIM*, из *XBALL*. Координату *y* можно найти прибавлением величины (3 + *SPEED*), умноженной на синус *RADAIM*, к *YBALL*.

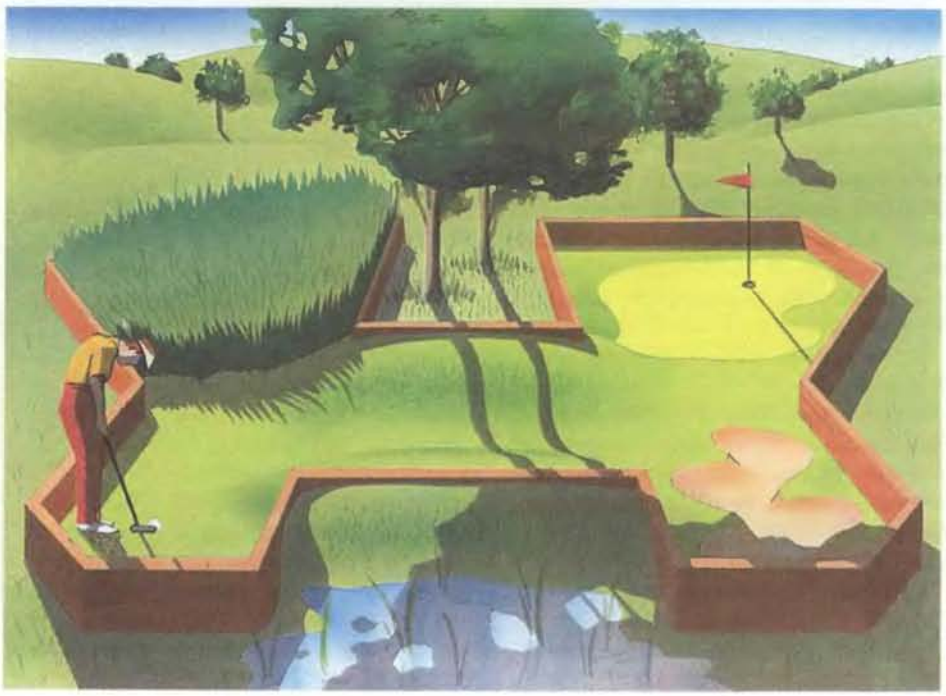
Чтобы определить угол наклона биты, потребуются дополнительные тригонометрические выкладки. Поскольку длина биты равна 16 пикселам, ее концевые точки удалены на расстояние 8 пикселов от центра. Таким образом, расстояние по оси *x* должно составлять 8 × cos (*RADAIM*), а по оси *y* расстояние равно 8 × sin (*RADAIM*).

Эти величины, обозначенные соответственно через *A, B, C* и *D*, вычисляются по отдельности в представленном ниже программном фрагменте, в строках со 140-й по 220-ю. Затем вычисляются действительные координаты «верхнего» конца биты (*TOPX, TOPY*) и ее «нижнего» конца (*BOTX, BOTY*). Наконец, по инструкции в строке 220 на экране рисуется линия, соединяющая эти точки.

```
140 A = 8 * COS (RADAIM)
150 B = 8 * SIN (RADAIM)
160 C = (SPEED + 3) *
    COS (RADAIM)
170 D = (SPEED + 3) *
    SIN (RADAIM)
180 XTOP = XBALL - C + B
190 YTOP = YBALL + D + A
200 XBOT = XBALL - C - B
210 YBOT = YBALL + D - A
220 LINE (XTOP, YTOP) -
    (XBOT, YBOT)
```

Может показаться, что теперь программа готова войти в третью фазу, а именно — изображение движущегося к цели мяча. Но, возможно, мы еще что-то не учли? А что, если игрок решит, будто неправильно задал угол наклона биты? Человеку ведь свойственно ошибаться, и ему следовало бы предоставить еще один шанс. Это можно сделать, предусмотрев по выбору играющего возможность перехода обратно к строке 10.

```
230 PRINT «ВВОДИТ 0 ДЛЯ УДА-
    РА»
240 INPUT PUTT
250 IF PUTT < > 0 THEN GOTO 10
```



Три препятствия в усложненном варианте игры в гольф

Здесь, если играющий введет любое другое число, кроме 0, программа снова перейдет к строке 10, где она очистит экран, перерисует заново игровое поле и опять предложит игроку задать новые значения для переменных *AIM* (цель) и *SPEED* (скорость). Если же игрок введет 0, то программа перейдет к заключительной фазе, которую мы еще в начале статьи описали следующим образом: «многократно продвинуть мяч (создать своеобразую мультипликацию по направлению и скорости биты)».

Однако, для того чтобы переместить мяч, программа должна изменить значения его координат *x* и *y* независимым образом. Для этого переменная *SPEED* раскладывается на две новые переменные: *XSPEED* — скорость в направлении *x* и *YSPEED* — скорость в направлении *y*.

```
260 XSPEED = SPEED *
    * COS (RADAIM)
270 YSPEED = SPEED *
    * SIN (RADAIM)
```

Заключительная часть программы представляет собой цикл, в котором многократно выполняются две операции. Мяч перемещается и затем происходит проверка, не попал ли он в лунку.

```
280 FOR K = 1 TO 300
290 XBALL = XBALL + XSPEED
300 YBALL = YBALL + YSPEED
310 CIRCLE (XBALL, YBALL), 4, 1
320 IF ABS (XBALL - 240) > 4
    THEN GOTO 360
330 IF ABS (YBALL - 100) > 4
    THEN GOTO 360
340 XSPEED = 0
350 YSPEED = 0
360 NEXT K
```

С помощью инструкции в строке 280 мы организуем простейший цикл. Переменная *K* отсчитывает от 1 до 300. Тем самым мы устанавливаем достаточное число повторений перемещения и изображения мяча, для того чтобы он достиг лунки, даже при скорости, равной 1. В теле цикла подсчитываются новые координаты мяча, а с помощью инструкции в строке 310 на экране строится его изображение.

В строках 320 и 330 производится покоординатная проверка с целью выяснить, не попал ли мяч в лунку. Если нет, программа сразу переходит к строке 360. Если да, то обе составляющие скорости мяча приравниваются нулю, и он застывает на месте.

Профессиональные программисты, возможно, сочтут неуместным применение в данном случае цикла типа FOR, потому что программа продолжает выполнение даже тогда, ког-

WWW.SCIAM-MAGAZINE.NAROD.RU

да мяч уже попал в лунку. Программа HOLE IN ONE продолжает проверять значения координат мяча до тех пор, пока индекс *K* не достигнет 300. Чтобы избавиться от этого недостатка, программист-любитель может добавить две инструкции в конец программы. Одна из них будет проверять, не равна ли скорость мяча 0. Если да, то вторая инструкция будет выводить какой-нибудь текст, сообщающий об успехе, типа: «Превосходный удар!» И наконец, последняя инструкция, требуемая в большинстве диалектов языка Бейсик, означает конец программы, END.

На этом и завершается программа HOLE IN ONE. Читатели, которые введут эту программу в компьютер и запустят ее, заметят одну любопытную особенность при попытках варьировать скорость мяча, задавая различные значения параметра *SPEED*. Благодаря необычной дискретной физике, законы которой, кажется, действуют на площадке для микрогольфа в программе HOLE IN ONE, мяч будет попадать в лунку лишь в том случае, если расстояние до лунки делится на скорость мяча без остатка!

То, что может показаться дефектом с одной точки зрения, возможно, покажется «интересным свойством» с другой: если физические законы становятся необычными случайно, то, наверное, их можно сделать такими и преднамеренно. В программе BIRDIE предусмотрены препятствия, каких не встретишь ни на одном травяном поле. Вдобавок здесь мяч может отскакивать от краев игрового поля.

Чтобы описать чудеса такого гольфа, я позволю себе отвлечься от конкретных свойств персонального компьютера IBM и языка Бейсик и вернуться к абстрактному языку алгоритмов. Площадка для гольфа в программе BIRDIE имеет Г-образную форму. Каким образом программа следит за тем, чтобы мяч все время оставался в пределах площадки? Даже в необычном мире микрогольфа угол падения должен оставаться равным углу отражения. Это правило нетрудно установить для шести отрезков прямых, не слишком заботясь о том, какой из этих отрезков обозначает ту или иную стенку; обозначим их *wall*1... *wall*6. Каждый раз, когда в цикле вывода изображения вычисляется новая позиция мяча, программа проверяет, не была ли пересечена какая-нибудь из стенок. Например, если была пересечена стенка *wall*1, мячик должен отразиться от нее.

Когда проверка и (возможно) результирующее отражение выполнены, программа производит проверку относительно стенки *wall*2, затем *wall*3 и т. д. Здесь возникает интересная проблема. Что, если мяч, «столкнувшись» с одной стенкой и отскочив от нее, пересечет и другую стенку? Не возникает ли такая ситуация, при которой мяч, ударяющийся о стенку вблизи угла, отскочит лишь от одной стенки, вместо того чтобы отскочить еще и от другой? Читатели, которые внимательно обдумают эту ситуацию, сумеют найти правильное решение.

После того как были выполнены все операции, связанные с возможными соударениями мяча на данном шаге цикла, программа BIRDIE выводит на экран его изображение и проверяет, не оказался ли он в лунке, точно так же, как это делала программа HOLE IN ONE. Ну а что, если неудачливый игрок промахнулся мимо цели? Мяч будет, отскакивая от стенок, снова и снова пересекать площадку, пока либо не свалится в лунку, либо не войдет в бесконечный цикл повторяющихся отскоков. Возможно, найдутся простаки, которые придут в восторг от такой перспективы, другие же раскритикуют программу за отсутствие реализма. И действительно, в игре отсутствует важнейший физический фактор — трение!

Чтобы движение мяча постепенно замедлялось, программа умножает переменную *SPEED* на некоторое постоянное число, меньшее единицы, скажем на 0.995, на каждом шаге цикла перемещения и проверок. Переменные *XSPEED* и *YSPEED* также умножаются на эту константу.

В конечном итоге скорость мяча может упасть до нуля, даже тогда, когда он так и не попал в лунку. В этом случае программа должна прекратить цикл, поэтому здесь вместо цикла типа FOR следует воспользоваться циклом типа WHILE. Цикл второго типа выполняется, пока не будет нарушено указанное условие. В нашем случае условие заключается в том, чтобы значение *SPEED* превосходило некоторое небольшое число, скажем 0.05. Когда скорость снижается до этой величины, программа BIRDIE возвращает играющего к интерактивной фазе, в которой он должен задать параметры для очередного удара.

С какими препятствиями может столкнуться гольфер в этой версии игры? Первое препятствие представляет собой круговую зону, или ловушку. Войдя в нее, мяч претерпевает случайные изменения в направлении и скорости своего движения. Программа BIRDIE проверяет, не находится ли мяч в данный момент в этой заколдованной зоне, и если находится, то она изменяет угол *RADAIM* на случайное число в диапазоне от −10 до +10. Тщательно спланировав несколько отскоков, хороший игрок может послать мяч в обход этого коварного круга.

Труднее миновать второе препятствие. Вокруг лунки, на некотором фиксированном расстоянии от нее, постоянно движется мяч-демон. Период его движения по орбите равен 10 шагам цикла программы. Если мяч игрока коснется демона, то будет вновь отброшен на исходную линию. Большинство читателей, до сих пор внимательно следивших за изложением, вероятно, сумеют найти способ избежать этого внушающего страх и ужас столкновения.

У программы EAGLE, как это видно из рисунка на с. 88, игровое поле сложнее по сравнению с BIRDIE. Хотя в основном она базируется на версии BIRDIE, у нее более развитая система препятствий. Например, всякий раз, когда мяч проходит через «горлышко», связывающее одну часть площадки с другой, он испытывает на себе силу, притягивающую его к центру прохода. Сила эта пропорциональна расстоянию мяча от центральной линии, как будто проход плавно сужается к центру. Есть тут и песчаная ловушка, в которой скорость мяча *SPEED* уменьшается в два раза, а не на 0.995, как на остальной части игрового поля. Только очень точным ударом можно обойти ловушку, чтобы не заставить мяч бесцельно метаться по площадке. И наконец, последнее препятствие, столкнувшись с которым мяч бесследно теряется из виду, даже не остановившись. Как найти его вновь? Если переместить биту в правильное положение, около нее появится мяч.

Мысль отправить читателей на экскурсию по зеленым лужайкам микрогольфа пришла мне после того, как я познакомился с игрой под названием «Зану — Гольф», недавно выпущенной фирмой Electronic Arts в Сан-Матео (шт. Калифорния). Можно, конечно, купить готовую игру, но при этом вы лишитесь удовольствия создать свою собственную. Во всяком случае, совсем не обязательно ограничиваться гольфом. Описанные здесь приемы можно легко приспособить и к другим играм, таким как электронный бильярд или микропинбол (электронный вариант пинбола — игры с наклонной доской со штырьками и лунками, по которой скатывается шарик, и сталкиваясь со штырьками, попадает в одну из лунок).

ЛЮБИТЕЛИ множества Мандельброта, поклонники фракталов и многие другие читатели с большим интересом прочли нашу статью, опубликованную в сентябрьском номере журнала, где речь шла о биоморфах, улитках и фрактальном попкор-

WWW.SCIAM-MAGAZINE.NAROD.RU

не. Творцом этого широкого разнообразия структур был К. Пикоувер из Исследовательского центра им. Томаса Уотсона фирмы IBM в Йорктаун-Хейтсе (шт. Нью-Йорк). Биоморфов плодит программа, основанная, по существу, на том же уравнении, которое генерирует множества Жюлиа (см. статью в рубрике «Занимательный компьютер» в январском номере журнала за 1988 г.). За счет введения дополнительных условий в итерационный процесс возникают изображения, напоминающие видимые в микроскоп живые организмы.

Д. Хоффман из Гринвилла (шт. Техас) отмечает, что биоморфы, благодаря тому что в формулах содержатся более высокие степени z по сравнению с формулами множеств Жюлиа, генерируются быстрее. Он рассматривает биоморфы как множества Жюлиа с «небольшим количеством» дополнительных элементов. Дополнительные пикселы встречаются не только в шипах и ресничках, но и в самом теле клетки: «Центральная часть биоморфа представляет собой либо связное множество Жюлиа, либо фрактальную пыль от взорвавшегося множества Жюлиа».

Несколько читателей столкнулись с некоторыми трудностями, пытаясь воспроизвести радиолярию с 12 шипами, изображенную на с. 80. Им удалось получить лишь фрагмент радиолярии, увеличенную часть тела клетки. Дело в том, что окно было задано слишком маленьким. Мы советуем читателям воспользоваться вместо окна 3 × 3 окном 20 × 20.

Неменьший интерес был проявлен и к плиткам Трюше, которые представляют собой квадраты с двумя круговыми дугами, как бы срезающими два противоположных угла. Любой массив из нулей и единиц можно преобразовать в причудливый плоский орнамент из кривых линий, если заменить каждый ноль плиткой с одной ориентацией рисунка и каждую единицу — плиткой с другой ориентацией. Пикоувер отмечает, что даже небольшое отклонение в последовательности битов от случайного распределения приводит к возникновению легко различимых закономерностей в соответствующем орнаменте из плиток. Этот факт может оказаться полезным для тех, кто занимается поиском закономерностей в данных.

Дж. Саверино из Трекслертауна (шт. Пенсильвания) — инженер-химик, обладающий художественными наклонностями, варьируя орнаменты из плиток, получил интересные результаты. Его произведения напоминают то какие-то странные городские планы, то несколько хаотические мавританские и византийские орнаменты. Наполняя некоторые связные области цветом, Саверино получает возможность быстро выбирать связные области в своих орнаментах.

Д. Спенсер из Валенсии (шт. Калифорния) предложил программу, создающую на экране своеобразную мультипликацию: после того как все биты массива были превращены в узор из плиток, программа начинает переключать по одному случайно выбираемые биты и изменять ориентации соответствующих плиток. По словам Спенсера, такая программа приводит весь экран в движение и получается захватывающее зрелище.

Наука и общество

Фотоника стучится в дверь

НАКОНЕЦ-ТО волоконно-оптические кабели приходят в наши дома. В последние годы такие кабели были проложены через континенты и по дну океанов, чтобы по ним передавать данные на большие расстояния. Местные телефонные компании с помощью таких же кабелей подключились к центральным коммутаторам. Теперь телефонные компании совместно с производителями оборудования связи работают над тем, чтобы протянуть волоконно-оптические кабели непосредственно к дверям потребителей.

Идея «глобальной деревни» М. Мак-Льюэна ждет реализации. По оптическим линиям связи, способным передавать намного больше информации, чем обычные провода, в наши дома будет поступать значительно больше видео- и звуковой информации. Для производителей оборудования электросвязи повсеместное использование оптических волокон открывает новый огромный рынок сбыта; по данным Национального научно-исследовательского совета, в США он составит не менее 100 млрд. долл. Однако для «глобальных деревень» нужны дешевые и надежные элементы фотоники (основа систем оптической связи), которые придут на смену традиционным линиям электросвязи. К ним относятся волоконно оптические кабели, лазерные передатчики и приемники, устройства для плотной записи и воспроизведения сигналов и т. д.

Однако, как отмечает С. Персоник из исследовательского центра Bell Communications Research, который участвовал в подготовке доклада по фотонике Национального научно-исследовательского совета США, американские компании, включая AT&T, все еще отстают в производстве новой техники. В результате этого отставания им придется в нелегкой борьбе отстаивать свою долю на рынке сбыта. «Есть пять или шесть крупных японских компаний, которые пытаются завоевать рынок сбыта для волоконно-оптической потребительской техники, — заявил Д. Ланг, директор AT&T Bell Laboratories. — И мы не можем догонять Японию без помощи со стороны».

Никто не оспаривает достижений США в исследованиях в области фотоники. Американским инженерам принадлежат многие изобретения на ранней стадии развития этой технологии и важные патенты в области лазерной техники и оптических волокон. Проводимые в настоящее время в США исследования — одни из лучших в мире. Прошлым летом на конференции в Японии группа инженеров из Bell Communications Research и AT&T Bell Laboratories поразила своих иностранных коллег, продемонстрировав миниатюрную подложку, содержащую два миллиона лазеров. Хотя это и чисто экспериментальное достижение, технология производства таких микросхем может оказаться важным шагом на пути к достижению большей интеграции электронных и оптических элементов.

В последние годы компания AT&T на основе собственных достижений в области волоконной оптики сумела создать мощное производство оборудования для дальней связи. На территории США компания проложила несколько волоконно-оптических кабельных линий дальней связи, а также выиграла конкурсный контракт на прокладку в начале 90-х годов второй линии оптических кабелей по дну Атлантического и Тихого океанов. Компания NTT, японский гигант в области связи, также предпочла заключить контракт с AT&T на строительство в Японии сети дальней связи с очень высокой скоростью передачи (2,5 Гбит/с). В то же время AT&T испытывает более суровую конкуренцию в производстве других компонентов фотоники. «Буквально десятки производителей предлагают свое оборудование, — отмечает Б. Юдж, управляющий компании NYNEX. — Выиграет тот, кто сможет снизить стоимость».

Стоимость является главной проблемой в волоконно-оптической промышленности. До нынешнего года региональные компании корпорации Bell не торопились прокладывать волоконные линии связи к жилым домам, так как это было слишком дорого. Что же касается затрат на строи-